MySQL in combination with Django (MySQL database migrations)

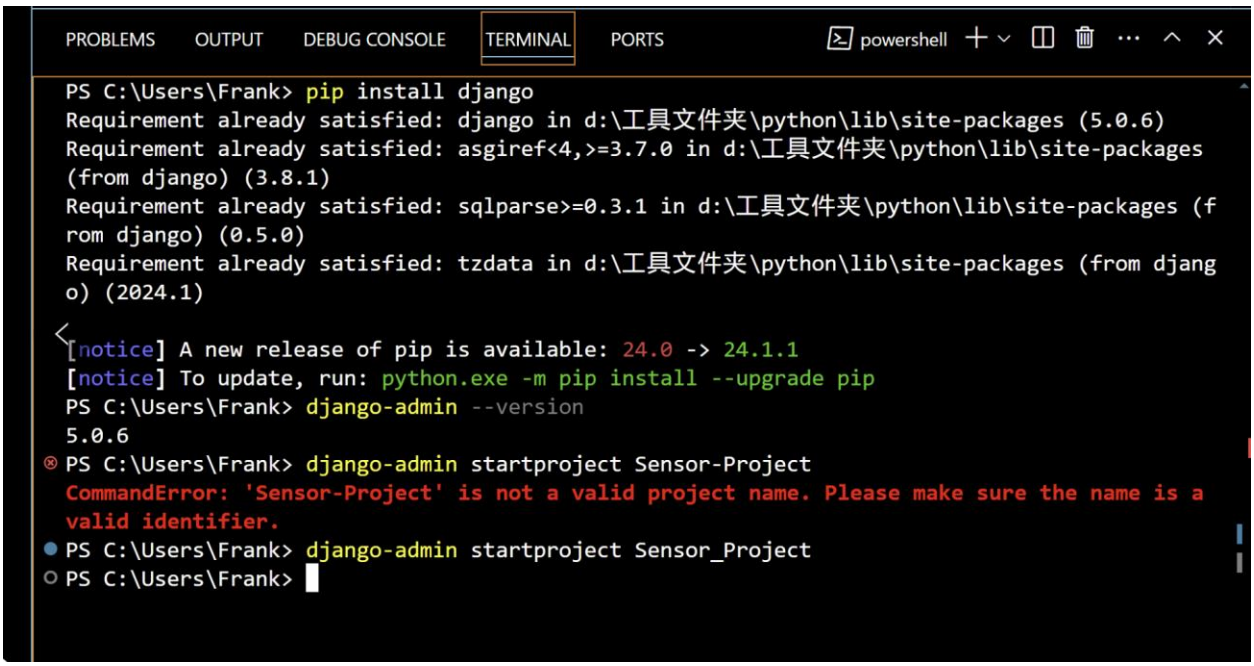
Key components of Django:

Model **View** **Template** **URLs**

Open the compiler to install Django by pressing Ctrl + ~ to open the terminal in the compiler.

Then enter the command **pip install django**, verify the installation by entering **django-admin --version**.

Next, start creating a Django project by entering **django-admin startproject Sensor_Project**.



Meaning of Django-Admin

The term "admin" refers to the administrator role, so actions need to be performed with administrator privileges, hence the use of django-admin .... Next, navigate to the directory: **cd Sensor_Project**. A project can contain multiple applications. An application is represented by the term "Application," abbreviated as "App." We can create an application called myapp.

Use the command **python manage.py startapp myapp** to create an application. The manage.py file is a key file for managing the entire Django project, and it is essential for running commands such as starting a project or application. It is automatically generated by Django.

Project Structure: The entire project contains several directories. Press Ctrl + B to open the side directory, it contains multiple Python files within myapp:

admin.py: for admin functionalities

models.py: for defining models

tests.py: for testing

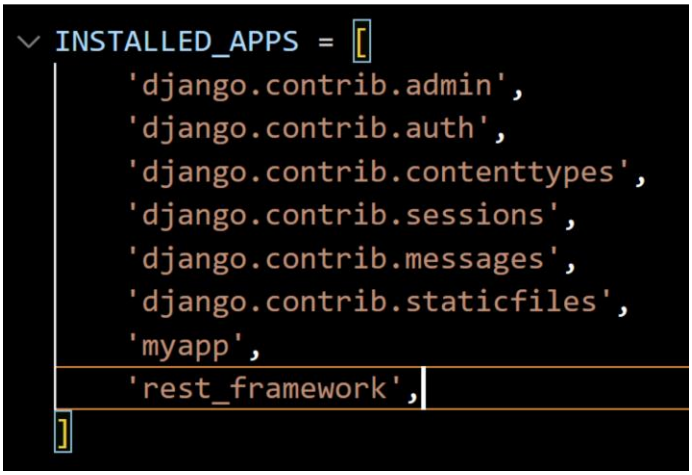
views.py: for defining views

Additionally, within the Sensor_Project directory, there are:

settings.py: for configuration settings

urls.py: for URL routing

First, click on setting, add myapp in installed_apps, and add the app you just created. Then fill in a 'rest_framework', add this first. Then, open views.py and add a simple view now.



views.py

```
from django.shortcuts import render

# Create your views here.

# Import the HttpResponse class from the django.http module, which is used to
generate HTTP responses.
from django.http import HttpResponse

def hello(request):
    return HttpResponse("Hello, world")
```

Write a simple hello world view, create a simple hello function, and the returned value is hello world.

Under myapp, create a new urls.py. The purpose of these few lines of code is to add the hello function in views to our url. Finally, add the following content in the project Sensor_Project/urls.py. This is a whole url equivalent to the nesting of urls.

myapp/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('',views.hello, name='hello'),
]
```

- **Import the path function from the django.urls module,** which is used to define URL patterns.
- **Import the views module under the current directory (usually the application directory)** so that the view functions in it can be referenced in the URL pattern.
- Define a list called urlpatterns, which contains URL patterns.
- In the urlpatterns list, a URL pattern is defined. **When a user visits the root URL (that is, the empty string ''), the hello view function in the views module will be called, and this URL pattern will be named hello.**

Sensor_Project/urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
]
```

- **Import admin from the django.contrib module to configure and access Django's admin backend.**
- **Import include and path functions from the django.urls module.** Path is used to define URL patterns and include is used to include other URL patterns.
- **Define a list called urlpatterns that contains URL patterns.**
- In the urlpatterns list, two URL patterns are defined:
    - ▪ When a user visits the admin/ URL, **admin.site.urls is called to handle the request. This is used by the Django admin.**
    - ▪ When a user visits the root URL (i.e. the empty string ''), the URL patterns from the myapp application are included. **This means that any request to the root**

URL will be handled according to the URL patterns in the myapp application.

Finally, test in the terminal and enter python manage.py runserver